ON FORECASTING DYNAMICS IN ONLINE DISCUSSION FORUMS

Chen Ling[*], *Di Cui* [*], *Guangmo Tong*[*] *and Jianming Zhu*[†]

[*]University of Delaware, USA, {cling, deechui, amotong}@udel.edu
[†]University of Chinese Academy of Sciences, China, jmzhu@ucas.ac.cn.

ABSTRACT

Online discussion forums are trending as popular platforms that allow asynchronous online interactions through a unique communication structure composed of main threads and the associated replies. In this paper, we present a learning framework – called SocialGrid – for modeling event dynamics in online discussion forums. Using a grid transformation, we explore the possibility of converting the problem of temporal space modeling into the problem of density space modeling. Inspired by the nature of the grid, we leverage a temporal convolution network to learn the dynamics in the density space. Changing the transformation precision, our approach can model the temporal dynamics at different granularities, thereby fulfilling prediction tasks with different needs. Experiments on real-world datasets have shown that our framework excels at various prediction tasks compared with other possible approaches.

Index Terms— information dynamics, online discussion forum, temporal convolution network

1. INTRODUCTION

Online discussion forums (ODFs) constitute a major branch of digital platforms, including social networking sites (e.g., Reddit and Digg), question-and-answer websites (e.g., Stack Overflow and Stack Exchange), and content sharing applications (e.g., Youtube and Pinterest). The study of ODFs are drawing increasing attention in recent years and has enabled important applications, such as thread recommendation in educational ODFs [1], latent community discovery [2], and user activity prediction [3]. Among the research advances, one of the fundamental problems is to model the temporal dynamics of the events occurring in ODFs.

Motivation. There have been plenty of learning methods that can deal with network-based platforms like Facebook and Twitter that are built on explicit user-user relationships (such as friendship relationship [4] and follower-followee relationship [5]). However, ODFs are distinct from the others by their unique *thread-reply* structure: people raise a discussion or a question by posting a thread, and users can respond to the threads by posting replies (Fig. 1). This unique structure opens new research opportunities in the sense that the existing

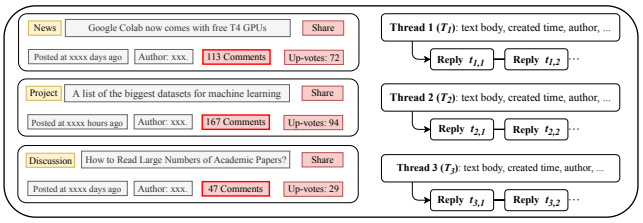
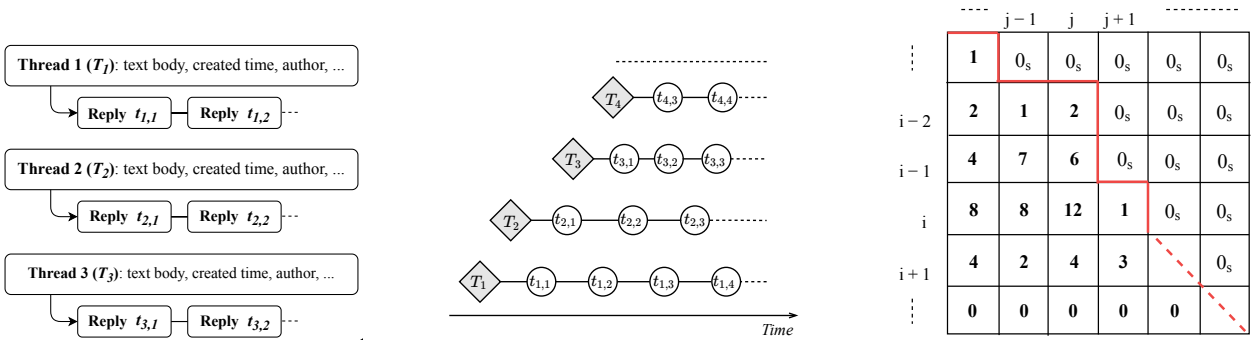


Fig. 1: The thread-reply structure of ODFs.

methods are not applicable. One challenge is that the dynamics under the thread-reply structure are less coherent without an explicit backbone network, and therefore, network-based models (such as the triggering models [6]) are not feasible. In addition, threads can be generated dynamically as new event streams, so one cannot use the temporal point models or deep learning methods that are designed for fixed-dimension sequences [7, 8]. In coping with these challenges, this paper seeks to design an end-to-end learning method suited for modeling threads and relies in ODFs.

Contribution. We propose a novel framework called *SocialGrid* for forecasting dynamic in ODFs, designed with two techniques: *grid transformation* and *temporal convolution network*. SocialGrid represents the thread-reply cascades using a grid structure where each cell summarizes the events within a short interval, which transforms the temporal point space into the event density space (Sec. 2). The obtained grid representation can approximate the original hypothesis space at different regularities, and it allows us to learn with matrix-like features, which are more tractable in terms of learnability. With the grid representation, we extend the one-dimension temporal convolution network (TCN) to a multi-dimension level and design models for predicting the arrival time of threads and replies at the grid level (Sec. 3). Our approach implicitly model the latent correlation between replies crossing different threads via 2D convolutions, and it therefore has the potential for resolving the cold start problem [9]: the initial replies in a new thread are hard to predict due to the lack of historical information. In experiments, we have found SocialGrid robust and promising for various prediction tasks (Sec. 4). In the supplementary materials, we provide a brief literature survey, the additional experimental results, and our source code.

978-1-6654-3864-3/21/$31.00 ©2021 IEEE



(a) Original View (b) Multi-sequence View (c) Grid View

Fig. 2: **Transformation Pipeline.** Fig. 2a shows a typical pattern of ODFs; Fig. 2b views the thread-reply dynamics as multiple time sequences; Fig. 2c illustrates the grid representation.

2. GRID TRANSFORMATION

An ODF (Fig. 2a) is composed of two types of event streams, thread stream $\{T_i\}$ and reply stream $\{t_{i,j}\}$, where each thread T_i is posted as a new web content to which users can respond through the associated replies $\{t_{i,1}, t_{i,2}, ..., t_{i,j}, ...\}$. We slightly abuse the notation and use $T_i \in \mathbb{R}^+$ and $t_{i,j} \in \mathbb{R}^+$ to also denote the arrival time of the event. Taking a supervised learning perspective, the event forecasting problem is to learn a conditional distribution:

Problem 1 (**Event Forecasting**). Given the arrival times $\mathcal{E}_{\leq t}$ of the events before a time point $t \in \mathbb{R}^+$, we are interested in the conditional distribution $\Pr[\mathcal{E}_{>t}|\mathcal{E}_{\leq t}]$, where $\mathcal{E}_{>t}$ denotes the arrival times of the events occurring after t.

Viewing each thread with its replies as a time sequence (Fig. 2b), the space we are concerned with is similar to the multi-variant point process [8], with the distinction that its dimension is dynamic and not unknown in advance, which makes the classic Hawkes-like models designed for the fixed-dimension case inapplicable. Notice that predicting the arrival time is equivalent to predicting the event density – the intensity (frequency) of the events at each time instance [10]. Thus, by relaxing an infinitesimal instance as a time interval, we have a natural grid-shaped representation of the temporal pattern of the events in ODFs, which we call the *Grid*.

Definition 1 (**Grid**). Grid $\mathcal{G}$ (Fig. 2c) is a matrix-like structure in which each column j is associated with a main thread T_j and the rows denote the timeline, where each row i represents a time interval of a controllable length d. The Grid consists of two types of elements. For each $\mathcal{G}_{i,j}$, if the main thread T_j has already arrived in or before the i-th time interval, $\mathcal{G}_{i,j}$ denotes the total number of replies of thread T_j occurring within the i-th time interval; otherwise, $\mathcal{G}_{i,j}$ is marked by a special symbol 0_s. A binary mask matrix $\mathcal{M}$ with the same shape as $\mathcal{G}$ is used to record the positions of the special symbol 0_s:

$$\mathcal{M}_{i,j} = \begin{cases} 1 & \text{if } \mathcal{G}_{i,j} = 0_s, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In the example of Fig. 2c, supposing that d is five minutes, thread T_j arrives at the time interval indexed by $i-2$, and it draws two replies in the first five minutes and six relies in the next five minutes.

Under the grid approximation, Problem 1 can be accordingly transformed into the grid prediction:

Problem 2 (**Grid Prediction**). Given the Grid $\mathcal{G}$, we aim to learn the distribution $\Pr[\mathcal{G}_{\geq i,j}|\mathcal{G}_{<i,j}]$ where $\mathcal{G}_{\geq i,j}$ denotes the entries that are at or beyond the i-th time interval, and $\mathcal{G}_{<i,j}$ denotes the entries before the i-th time interval.

Learning the Grid enables us to forecast the number of events in the future time intervals, and we therefore can predict the arrival time of the upcoming events at different granularities by adjusting the interval length d used for building the Grid. The main merit of doing so is that, compared with temporal sequences, the grid representation is much easier to learn using machine learning methods. We can see that the grid recovers the event sequences when d approaches to 0. This creates an interesting trade-off controlled by d, which will be studied in experiments. In the next section, we discuss the learning strategy to solve Problem 2.

3. LEARNING STRATEGY

One can imagine the grid in Fig. 2c as a *social image* in which each cell provides the features summarizing the events within a small time period, making the convolutional neural networks a natural choice for mining the latent representation. Our approach employs a temporal convolution architecture to parameterize the distribution $\Pr[\mathcal{G}_{\geq i,j}|\mathcal{G}_{<i,j}]$.

We first design the individual modules and then present our model pipeline.

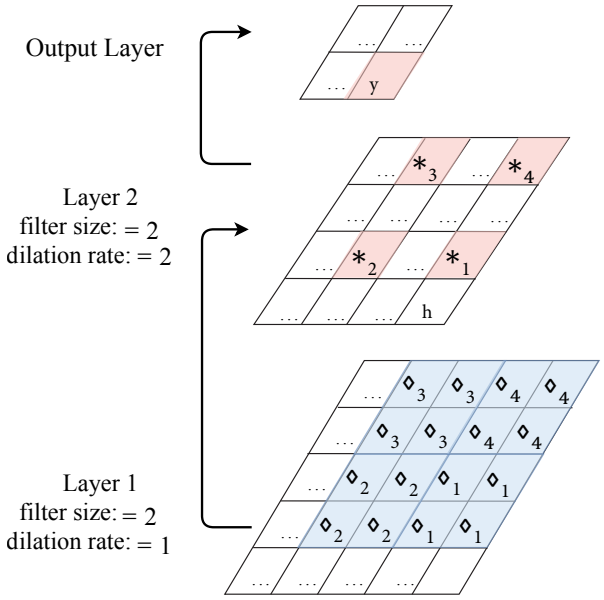


Fig. 3: **2D dilated causal convolution.** Suppose that the element x in the first layer corresponds to the element y in the output layer. As shown in the figure, the element y is decided by $*_i, i \in [1, 2, 3, 4]$ at Layer 2, and each of the $*_i$ element is decided by the $\Diamond$ element with the same index number. By stacking with more dilated 2D causal convolution layers, the output element y will be decided by a larger receptive field.

3.1. 2D Temporal Convolution Layer

On the basis of the temporal convolution network for modeling one-dimension sequences [11], we design the 2D temporal convolution layers through two techniques: causal convolution and dilated convolution.

Causal Convolution. For temporal event modeling, causal convolution ensures the temporal causality: the representation of a grid in $\mathcal{G}$ should be determined by the past grids with smaller row indices. Formally, the feature transformation between layers can be written as:

$$\Lambda^l_{i,j} = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \mathbf{f}_{k_1,k_2} \cdot \Lambda^{l-1}_{i-k_1,j-k_2},$$

where $\Lambda^l_{i,j}$ represents the hidden representation at the l-th layer (with Λ^0 being the input features), and $\mathbf{f} \in \mathbb{R}^{K\times K}$ is a square-shaped convolution filter with size $K \in \mathbb{Z}^+$.

Dilated Convolution. The traditional CNN often attempts to enlarge its receptive field by stacking layers, but this also unnecessarily increases the model complexity [11]. TCN employs the dilated convolution to exponentially enlarge the receptive field when stacking with more convolution layers. For the two-dimension case, the causal convolution endowed with a dilation rate τ is formally given by

$$\Lambda^l_{i,j} = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \mathbf{f}_{k_1,k_2} \cdot \Lambda^{l-1}_{i-k_1\cdot\tau,j-k_2\cdot\tau}.$$

Therefore, the recursive formula of the receptive field RF_l of the l-th layer is analytically given by

$$\mathrm{RF}_l = \Big((\mathrm{RF}_{l-1} - 1) + K + (K-1)\cdot(\tau-1)\Big)^2$$

with $\mathrm{RF}_1 = K^2$. As for our grid prediction problem (Fig. 2c), the receptive field indicates a) the number of the past threads to be considered (along the columns) and b) the number of the past time intervals to take effect (along the rows). Fig. 3 provides an illustration of the 2D dilated causal convolution layer.

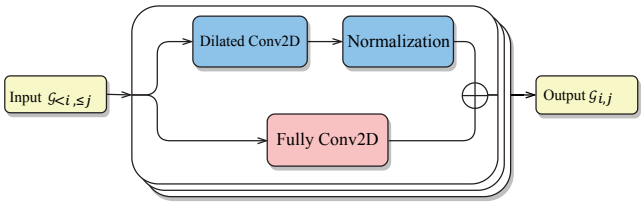


Fig. 4: **Temporal Convolution Block.**

3.2. Temporal Convolution Block

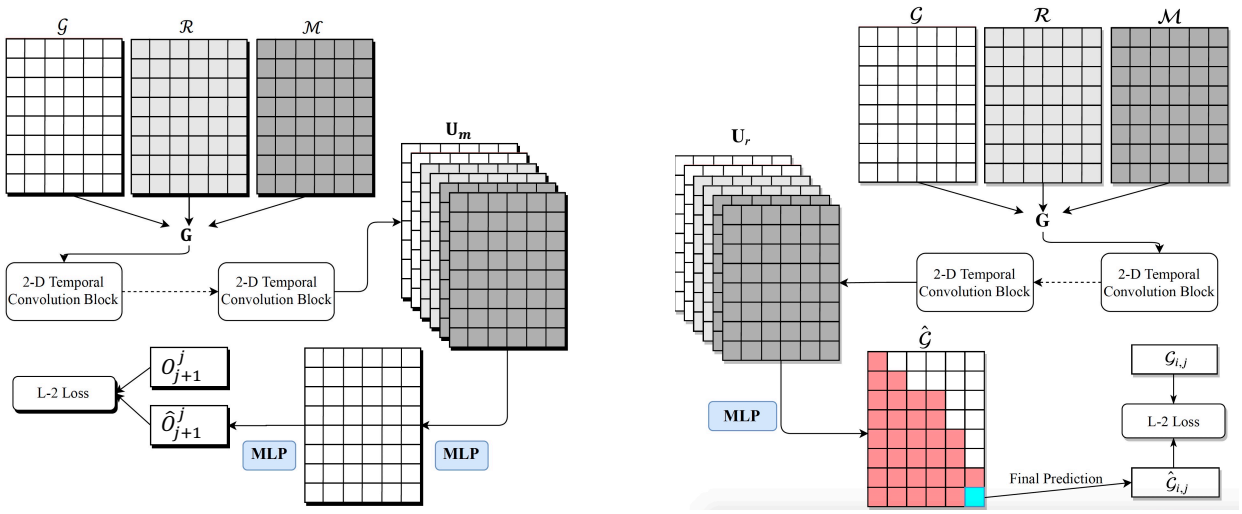
With the proposed 2D temporal convolutional layers, we design the temporal convolution block, which constitutes the main modules of our model pipeline. In each block, a 2D dilated causal convolution layer is followed by a normalization layer (denoted by Norm) and a non-linear activation function (denoted by σ) to reduce the internal covariate shift, which leads to the transformation

$$\Lambda^* = \sigma(\mathrm{Norm}(\Lambda^l)).$$

To reduce the potential risk of the network performance degeneration caused by the deep network structure, we add a residual connection [12] between the temporal convolution blocks. This is intended to help the model directly learn the modification from the identity mapping of the last block (instead of the full transformation). In our design, the prediction in the output $\Lambda^*_{i,j}$ is added element-wisely with the input $\Lambda^l_{i,j}$ from the last layer, and we apply an additional fully convolution network to Λ^* to account for discrepant shapes between Λ^* and Λ^l. As a result, the output Λ^{l+1} can be represented as

$$\Lambda^{l+1}_{i,j} = \sigma(\Lambda^*_{i,j} \oplus \Lambda^l_{i,j}).$$

The structure of the block is summarized in Fig. 4. We adopt batch normalization [13] and use the Parametric Rectified Linear Unit (PReLU) [14] as the activation function.



(a) Model of Main Thread

(b) Model of Future Replies

Fig. 5: **Model Pipeline.** Each model takes the tensor $\mathbf{G}$ consisting of three feature channels $\mathcal{G}$, $\mathcal{R}$ and $\mathcal{M}$, and the inputs are processed by a sequence of temporal convolution blocks to generate the output tensors $\mathbf{U}_m$ and $\mathbf{U}_r$. $\mathbf{U}_m$ and $\mathbf{U}_r$ are further compressed by MLP to get $\hat{O}^j_{j+1}$ and $\hat{\mathcal{G}}_{i,j}$, respectively, for predicting the thread arrival times and reply numbers in the grid. MSE loss between the predicted value and the ground truth is used for training the parameters. During the training, the entire grid $\mathbf{G}$ is slices into 200×200 subareas as training samples, though the grid $\mathcal{G}$ is conceptually infinite.

3.3. Model Pipelines

For enhancing the grid prediction (Problem 2), we utilize two auxiliary features, in addition to the grid input $\mathcal{G}$. The first feature matrix (denoted by $\mathcal{R}_{i,j}$) records the number of time intervals (rows) from each cell $\mathcal{G}_{i,j}$ to the first time interval of the associated main thread T_j. We code $\mathcal{R}_{i,j}$ as zero if $\mathcal{G}_{i,j}$ is the special symbol 0_s, and further normalize all the columns in $\mathcal{R}$ to the range of $[0, 1]$ in order to keep the count of each cascade in the same scale. $\mathcal{R}$ is intended to inform our model of the latent influence from the associated main thread. We use the mask matrix $\mathcal{M}$ (Eq. 1) as the second feature matrix to remind the model of the locations of the special symbol.

Under the grid presentation, the arrival time of future main threads can be predicted by inferring $O^j_{j+1} \in \mathbb{N}$, which is defined to be the number of intervals (rows) between the first intervals of two consecutive main threads T_j and T_{j+1}. Successfully doing so will allow us to (approximately) estimate the arrival time T_{j+1} by

$$T_{j+1} = T_j + (O^j_{j+1} \cdot d).$$

To this end, we parameterize $\Pr[O^j_{j+1} | \mathcal{G}_{\leq i, \leq j}]$ using a sequence of temporal convolutional blocks, where the residue connection is omitted and the number of blocks is taken as a hyperparameter determined by the grid search. To predict the number of future replies $\mathcal{G}_{\geq i,j}$ at the grid level, the distribution $\Pr[\mathcal{G}_{\geq i,j} | \mathcal{G}_{<i,j}]$ is again learned through temporal convolutional blocks. The entire process is detailed in Fig. 5.

4. EXPERIMENT

4.1. Experiment Setting

We have implemented several standard and popular methods for sequence modeling, including ARIMA [15], GRU [16] and LSTM [7]. Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are adopted to measure the difference between the ground truth array and the predicted array. They are measured in hours in predicting the arrival times and measured in number in predicting the number of replies.

We adopt two Reddit datasets: NBA and NFL. NBA dataset contains $1,610$ main threads and $71,638$ corresponding replies collected from the NBA subreddit during the 2019 playoff month. We use the data of the first three weeks as the training-validation set, and use the rest of the data as the testing set. NFL dataset is created by retrieving the discussions on Superbowl 2019 during the week of the NFL final match. This dataset contains $1,895$ main threads and $138,911$ associated replies. We use the data in the first five days (roughly $1,600$ thread-reply cascades) as the training-validation set, and use those in the rest two days as the testing set.

In the rest of this section, we present the results on two tasks: non-adaptive prediction and breakout thread detection. The supplementary material contains the full experiment, including the detailed implementation of each method, results on adaptive-prediction and ablation study, results of more competitors, and the source code.

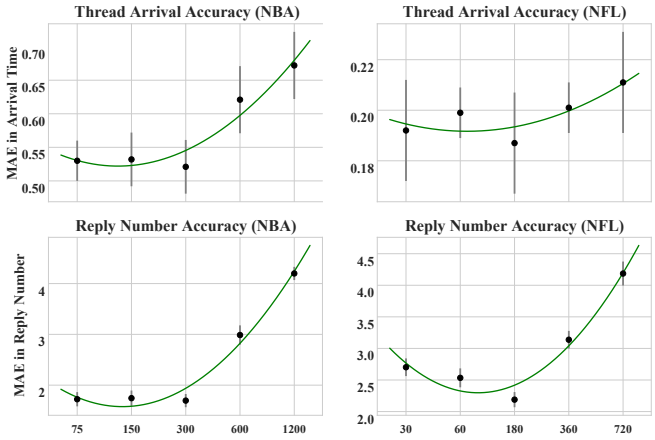


Fig. 6: **Sensitivity Analysis.** The x-axle denotes the value of d in seconds, and the y-axle denotes the prediction error.

4.2. Non-adaptive Prediction

Given the current snapshot of the grid $\mathcal{G}_{\leq i,\leq j}$, the first task we consider is to predict the arrival time of the next thread (i.e., T_{j+1}) as well as the number of replies occurring in the next time interval (i.e., row $i+1$). This prediction task is non-adaptive in the sense that the predictions are made using only the given histories, as opposed to the adaptive prediction where the prediction of distant future events relies on the predicted events in the near future. We first present the sensitivity study to determine the value of d used in SocialGrid and then examine the performance of the considered approaches.

Sensitivity Analysis. The value of d affects the model precision by controlling the minimum time interval that SocialGrid can model. There is an essential trade-off: the larger d decreases the model precision and therefore makes the prediction coarse, but this also reduces the model complexity, making it easy to learn. Thus, we examine the impact of d on the final metric (MAE). Fig. 6 shows the performance of SocialGrid with different values of d in predicting the thread arrival times and reply numbers. Interestingly, in all four experiments on the two datasets, the MAE curve fits a convex function, and there exists an optimal length d that gives the best prediction accuracy, which echos the aforementioned trade-off – extreme values are not preferred. The optimal d is reached at 300 seconds for the NBA dataset and at 180 seconds for the NFL dataset. In our experiments, we fix d as 300 and 180 seconds for the NBA dataset and the NFL dataset, respectively.

Main Thread Arrival Time. We first consider the task of predicting the arrival time of the future main threads. The main thread stream is taken as a one-dimension sequence so that non-SocialGrid approaches can be readily applied. We predict the arrival times of 100 randomly selected main threads for each dataset and report the average MAE and RMSE in hours, together with the standard deviation (Table 1). As shown in Table 1, SocialGrid consistently outperforms other approaches. Plausibly, ARIMA is limited by its model capacity and RNN-based models (GRU and LSTM) cannot utilize the information from the associated replies. In contrast, SocialGrid can capture the thread-reply dependency due to the 2D convolutional operations.

Model	NBA Dataset		NFL Dataset	
	MAE	RMSE	MAE	RMSE
ARIMA	1.35 $_{0.21}$	1.88 $_{0.34}$	0.59 $_{0.09}$	0.75 $_{0.07}$
GRU	1.24 $_{0.09}$	1.84 $_{0.09}$	0.42 $_{0.06}$	0.61 $_{0.05}$
LSTM	1.73 $_{0.10}$	2.62 $_{0.19}$	0.51 $_{0.05}$	0.78 $_{0.06}$
SocialGrid	0.58 $_{0.04}$	0.93 $_{0.05}$	0.18 $_{0.03}$	0.28 $_{0.04}$

Table 1: **Main Thread Arrival Time Prediction.**

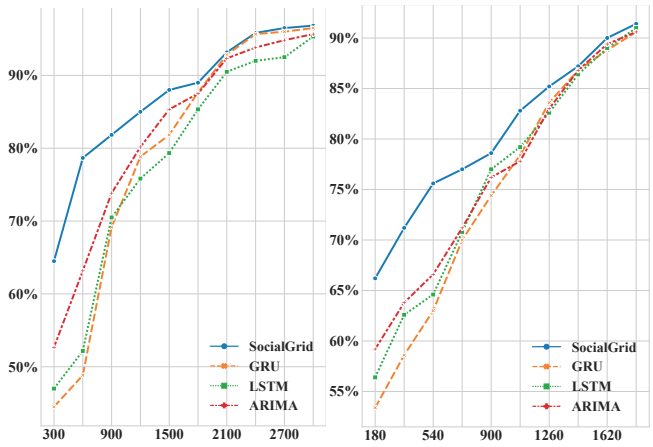
Model	NBA Dataset		NFL Dataset	
	MAE	RMSE	MAE	RMSE
ARIMA	5.05 $_{0.22}$	6.27 $_{0.24}$	2.56 $_{0.02}$	2.90 $_{0.03}$
GRU	4.82 $_{0.21}$	5.90 $_{0.20}$	2.21 $_{0.07}$	2.81 $_{0.08}$
LSTM	4.63 $_{0.29}$	5.80 $_{0.41}$	2.24 $_{0.12}$	2.86 $_{0.04}$
SocialGrid	1.97 $_{0.04}$	3.81 $_{0.16}$	1.52 $_{0.05}$	2.16 $_{0.10}$

Table 2: **Reply Number Prediction.**

Reply Number. When predicting the reply numbers in the next row, treating the reply stream of each thread as one point process suffers from the cold start issue: the first several replies are not predictable due to the lack of sufficient historical data. Therefore, we train the non-SocialGrid models using the arrival time information of $3,000$ replies from previous cascades to infer the average pattern of the replies, and use the obtained model to predict future reply numbers. The testing is done on 200 randomly selected thread-reply cascades, and for each of them we predict the reply numbers in 20 consecutive time intervals. For non-SocialGrid methods, we adaptively simulate the arrival time of future replies and count the predicted number within the next time interval, in order to compare them with SocialGrid. The results are shown in Table 2. As we can see from there, SocialGrid still has the lowest prediction error on both datasets, and its superiority is evident. The limitation of non-SocialGrid methods is obvious: since threads may have different evolution patterns, it is impossible to get accurate predictions using a fixed fitting function. Furthermore, compared with other methods, SocialGrid provides a more robust prediction (in terms of standard deviation).

4.3. Breakout Thread Detection

While we have shown that SocialGrid performs better than the baselines, we cannot conclude that the arrival time of each



(a) NBA (b) NFL

Fig. 7: **Breakout Threads Detection.** The x-axle denotes the length (sec) of the start duration, and the y-axle denotes the classification accuracy.

individual event has been accurately predicted. A less ambitious task is to predict macro-level quantities such as the total amount of the replies in one thread; one important application is to identify breakout threads, which is closely related to rumor detection and anomaly detection in ODFs. More specifically, we aim at forecasting if a thread would attract more than the average replies, by using the data in the first several time intervals. To this ends, we first find the average cascade size, and define the breakout threads as those having a total number of replies more than twice the average. Given the information in the first kd seconds (called start duration) of a thread, $k \in \{1, ..., 10\}$, we use the learned models to simulate the final volume of the replies, and report if the thread would be classified as an outbreak cascade. The classification accuracy is given in Fig 7. Intuitively, this task becomes easier when the information of a longer start duration is provided, so the accuracy would increase with k. As we can see from both figures, SocialGrid can identify more breakout threads in their early stages for both datasets. When the information of 300 seconds is provided, SocialGrid is able to identify roughly 65% of the breakout cascades in the NBA dataset, and an accuracy of 66% is observed on the NFL dataset with a start duration of 180 seconds. Other approaches often need more than 600 seconds to achieve the same.

5. CONCLUSION

In this paper, we propose a TCN-enhanced deep learning framework (SocialGrid) for modeling the information dynamics in ODFs. We propose to leverage a grid-shaped representation to reduce the model complexity for a better learning effect, and utilize the 2D temporal blocks to build a learning pipeline for modeling the obtained grid structure. Experiments have shown that our method outperforms alternative approaches on two real-world datasets.

6. REFERENCES

[1] A. S. Lan, J. C. Spencer, Z. Chen, C. G. Brinton, and M. Chiang, "Personalized thread recommendation for mooc discussion forums," in *ECML*, 2018.

[2] S. Liu, S. Yao, D. Liu, H. Shao, Y. Zhao, X. Fu, and T. Abdelzaher, "A latent hawkes process model for event clustering and temporal dynamics learning with applications in github," in *ICDCS*, 2019.

[3] C. Romero, M.-I. López, J.-M. Luna, and S. Ventura, "Predicting students' final performance from participation in on-line discussion forums," *Computers & Education*, 2013.

[4] S. Vallor, "Flourishing on facebook: virtue friendship & new social media," *Ethics and Information technology*, vol. 14, no. 3, pp. 185–199, 2012.

[5] T.-Q. Peng, M. Liu, Y. Wu, and S. Liu, "Follower-followee network, communication networks, and vote agreement of the us members of congress," *Communication research*, vol. 43, no. 7, pp. 996–1024, 2016.

[6] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *SIGKDD*, 2003.

[7] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, 2002.

[8] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *SIGKDD*, 2015.

[9] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *SIGIR*, 2002.

[10] G. Last and M. Penrose, *Lectures on the Poisson process*. Cambridge University Press, 2017, vol. 7.

[11] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv:1502.03167*, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015.

[15] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "Arima models to predict next-day electricity prices," *IEEE transactions on power systems*, 2003.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv:1412.3555*, 2014.

ON FORECASTING DYNAMICS IN ONLINE DISCUSSION FORUMS (SUPPLEMENTARY MATERIAL)

Chen Ling[*]*, Di Cui* [*]*, Guangmo Tong*[*] *and Jianming Zhu*[†]

[*]University of Delaware, USA, {cling, deechui, amotong}@udel.edu
[†]University of Chinese Academy of Sciences, China, jmzhu@ucas.ac.cn.

A. RELATED WORK

The current research on ODFs is ranging from user behavior modeling [1,2] to content analysis and recommendation [3,4], focusing on the Massive Open Online Course platforms. Very few works have paid attention to the unique structure of the ODF in studying the information diffusion dynamics [3,5–7]. In the study of general social networks, existing works [8–10] have utilized the Gaussian-based point processes [11] to predict the popularity of a post by fitting a single intensity function. Deep recurrent architectures [12, 13] have also been leveraged for mining the complex latent relationship between the temporal sequences. Nevertheless, the current temporal point process models and RNN models are often limited to the fixed-dimension cases, while ODFs are composed of dynamic thread-reply cascades. Convolution networks are proven to have the ability to model sequence data back to the 90s [14, 15]; over time, CNNs have further been utilized in various sequence modeling tasks (e.g., [16, 17]) with competitive results comparing with deep recurrent architectures. To date, inspired by the dilated convolution in [18], TCN [19] has been introduced as one of the strong competitors in various sequence modeling tasks. The proposed framework SocialGrid is specially designed based on TCN to process the thread-reply structure of ODFs.

B. EXPERIMENT (EXTENDED VERSION)

B.1. Experiment Setting

We implement SocialGrid with Tensorflow and employ Adam optimizer with a learning rate of 1.0×10^{-3} for training. In addition, L2 regularizer (1.0×10^{-2}) is applied to all convolution filters in the model of predicting future replies. We execute the grid search strategy to locate the best parameters through validation set: the number of filters is chosen from $\{16, 32, 64, 128\}$, and the filter size is chosen from $\{3, 5, 7, 9\}$ in all dilated causal convolution layers. The number of temporal convolution blocks is taken from $\{3, 4, 5, 6, 7\}$, and the dilation rate τ is 2^{i-1} at the i-th block.

While Grid $\mathcal{G}$ is conceptually an infinite matrix, it is sufficient for the training purpose that we slice the entire Grid $\mathcal{G}$ into segments that cover the receptive field of the predicted cell. The segments of feature matrices $\mathcal{R}$ and $\mathcal{M}$ are generated accordingly. We apply zero-padding to the top and left side of the feature matrices to ensure the input-output shape consistency and to achieve the temporal causality.

Baselines. We compare our framework SocialGrid with four types of baselines:

- Traditional Statistical Model. ARIMA (Auto Regressive Integrated Moving Average) [20] is the classic statistical model for time-series analysis. We implement the ARIMA model with the public Python library [21].
- Recurrent Neural Networks (RNN). Recurrent architectures are standard deep learning methods for sequence modeling [22, 23]. We implement the vanilla GRU and LSTM as the comparison model, where both GRU and LSTM contain 64 hidden units and 4 layers of the recurrent structure.
- Temporal Point Processes (TPP). Temporal point process [11] is another popular class for modeling temporal sequences, and a number of following works [8, 10] have utilized the TPP to study online social network. We select the most fundamental TPP model Hawkes process [11] as well as a recent TPP model - NesTPP [7] that is specified in simultaneously modeling the diffusion of main threads and associated replies. We use the public Python library [24] for implementation.
- Variants of SocialGrid. We further compare SocialGrid with its variants. Specifically, one-dimension TCN [19] is implemented to examine if building multi-dimension temporal convolution structure can benefit the prediction accuracy. In order to investigate the efficacy of the channels, we test SocialGrid-S which is the SocialGrid with only one feature channel $\mathcal{G}$, and SocialGrid-M which makes use of two feature channels of $\mathcal{G}$ and $\mathcal{R}$ without the mask $\mathcal{M}$.

Evaluation Metrics. Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are adopted to measure the error between the ground truth array and the predicted array. They are measured in hours when we predict the arrival times

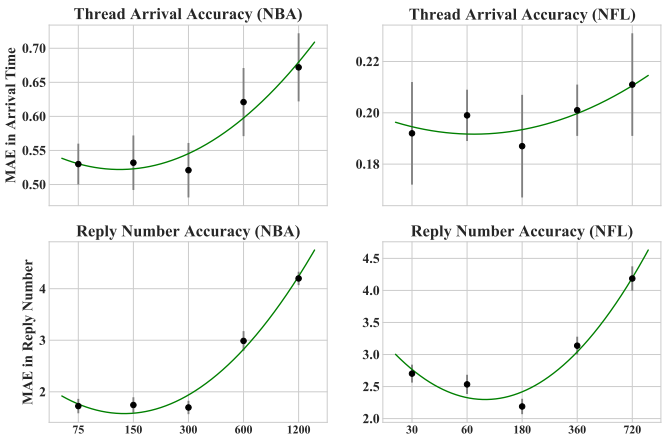


Fig. 1: **Sensitivity Analysis.** The x-axle denotes the value of d measured in seconds, and the y-axle denotes the prediction error.

Model	NBA Dataset		NFL Dataset	
	MAE	RMSE	MAE	RMSE
ARIMA	$1.35_{0.21}$	$1.88_{0.34}$	$0.59_{0.09}$	$0.75_{0.07}$
GRU	$1.24_{0.09}$	$1.84_{0.09}$	$0.42_{0.06}$	$0.61_{0.05}$
LSTM	$1.73_{0.10}$	$2.62_{0.19}$	$0.51_{0.05}$	$0.78_{0.06}$
TCN	$0.88_{0.10}$	$1.13_{0.12}$	$0.35_{0.08}$	$0.70_{0.12}$
SocialGrid-S	$0.59_{0.03}$	$1.11_{0.04}$	$0.21_{0.03}$	$0.30_{0.02}$
SocialGrid-M	$0.52_{0.03}$	$0.92_{0.07}$	$0.17_{0.01}$	$0.27_{0.02}$
SocialGrid	$0.58_{0.04}$	$0.93_{0.05}$	$0.18_{0.03}$	$0.28_{0.04}$

Table 1: **Main Thread Arrival Time Prediction.**

and measured in number when we predict the amount of the replies.

Dataset. We adopt datasets of Reddit, which is one of the largest ODFs. The underlying assumption behind our modeling approach is that the events are correlated and mutually excited so that they are learnable; therefore, we assemble the threads concerning one topic as one dataset. In particular, we target two popular subreddits: NBA and NFL.

NBA Dataset [7]. This dataset is composed of $1,610$ main threads and $71,638$ corresponding replies related to the famous player LeBron James, collected from the NBA subreddit of one month during the 2019 Playoffs. We apply the main threads and linked replies in the first three weeks as the training-validation set, and use the rest of the data as the testing set.

NFL Dataset. We retrieved the discussions of the topic *Superbowl 2019* during the week of the NFL final match. This dataset contains $1,895$ main threads and $138,911$ associated replies. We use the threads and their replies arriving in the first 5 days (roughly $1,600$ thread-reply cascades) as the training-validation set, and those in the rest two days serve as the testing set.

B.2. Application #1: Non-adaptive Prediction

For an anchor cell $\mathcal{G}_{i,j}$, the first task we consider is to predict the arrival time of the next thread (i.e., T_{j+1}) and to predict the number of replies occurring in the next time interval (i.e., row $i+1$). This prediction task is non-adaptive since the predictions are made using only the given histories, as opposed to the adaptive prediction where the prediction of distant future events relies on the predicted events in the near future. We first present the sensitivity study to determine the value of d used in SocialGrid and then examine the performance of the considered approaches.

Sensitivity Analysis. The value of d affects the model precision by controlling the minimum time interval that can be modeled by SocialGrid. There is an essential trade-off: the larger d decreases the model precision, making the prediction coarse, but it also reduces the model complexity, making it easy to learn. Therefore, we are interested in the impact of d on the final metric (MAE). Fig. 1 shows the performance of SocialGrid with different values of d in predicting the thread arrival times and reply numbers. Interestingly, in all four experiments on the two datasets, the MAE curves fit a convex function, and there exists an optimal length d that gives the best prediction accuracy, which echos the aforementioned trade-off. The optimal d is reached at 300 seconds for the NBA dataset and at 180 seconds for the NFL dataset. In the rest experiments, we fix d as 300 and 180 seconds for the NBA dataset and the NFL dataset, respectively.

Main Thread Arrival Time. We first consider the task of predicting the arrival time of the future main threads. The main thread stream is taken as a one-dimension sequence so that non-SocialGrid approaches can be readily applied. We predict the arrival times of 100 randomly selected main threads for each dataset and report the average MAE and RMSE in hours, together with the standard deviation (Table 1). As shown in Table 1, TCN-related models consistently outperform other approaches, and SocialGrid-M achieves the best performance. ARIMA is limited by its model complexity; RNN-based models and TCN can capture a more complex underlying relation, but they cannot utilize the information from the associated replies. Comparing TCN with other non-SocialGrid methods, we see that simplifying the learning task by using the grid structure is indeed helpful; comparing TCN with SocialGrid methods, it confirms that modeling the mutual influence between the thread and reply through 2D temporal convolutions further improves the generalization performance. For SocialGrid and its variants, we observe that SocialGrid and SocialGrid-M slightly outperform SocialGrid-S, which suggests that the feature $\mathcal{R}$ is relatively useful while $\mathcal{M}$ may have a negative effect.

Reply Number. While predicting the reply numbers in

Model	NBA Dataset		NFL Dataset	
	MAE	RMSE	MAE	RMSE
ARIMA	$5.05_{0.22}$	$6.27_{0.24}$	$2.56_{0.02}$	$2.90_{0.03}$
GRU	$4.82_{0.21}$	$5.90_{0.20}$	$2.21_{0.07}$	$2.81_{0.08}$
LSTM	$4.63_{0.29}$	$5.80_{0.41}$	$2.24_{0.12}$	$2.86_{0.04}$
TCN	$2.99_{0.14}$	$3.48_{0.31}$	$2.02_{0.10}$	$2.58_{0.15}$
SocialGrid-S	$2.20_{0.09}$	$3.86_{0.17}$	$1.63_{0.04}$	$2.13_{0.06}$
SocialGrid-M	$2.06_{0.07}$	$3.83_{0.27}$	$1.57_{0.03}$	$2.28_{0.07}$
SocialGrid	$1.97_{0.04}$	$3.81_{0.16}$	$1.52_{0.05}$	$2.16_{0.10}$

Table 2: **Reply Number Prediction.**

the next cell in $\mathcal{G}$, treating the reply stream of a thread as one point sequence suffers from the cold start issue: the first several replies are not predictable due to the lack of sufficient historical data. Therefore, we train the non-SocialGrid models using the arrival time information of $3,000$ replies from previous cascades to infer the average pattern of the replies, and use the obtained model to predict future reply numbers. The testing is done on 200 randomly selected thread-reply cascades, and for each of them we predict the reply numbers in 20 consecutive time intervals. For non-SocialGrid methods, we adaptively simulate the arrival time of future replies and count the predicted number within one time interval (cell). The results are shown in Table 2. As we can see from the table, TCN-based models still exhibit a lower prediction error on both datasets, and their superiority is evident. For non-SocialGrid methods that only utilize univariate thread-reply cascade information as their training instances, since each information cascade may have different evolution patterns, it is difficult for them to successfully learn all the reply evolutions using the same fitting function. Furthermore, compared with the TCN, SocialGrid considers the mutual influence between information cascades, which leads to a more robust prediction in terms of the standard deviation. In the ablation study, we do not see much difference between the three methods and observe that the two extra feature matrices $\mathcal{R}$ and $\mathcal{M}$ are both helpful.

B.3. Application #2: Adaptive Prediction

Given the full history $\mathcal{G}_{<i,\leq j}$, we can infer the arrival time of thread T_{j+1} by predicting O^j_{j+1}, and repeating this procedure, we are able to adaptively predict a sequence of future threads. Similarly, the reply streams can also be predicted adaptively using SocialGrid. We compare our model with Hawkes and NesTPP, omitting ARIMA and RNN-based methods since they have been proven to be ineffective for non-adaptive prediction. While using Hawkes model, we treat the thread and reply streams as two individual temporal point processes, and the two processes are trained separately with the information of $1,000$ main threads and $3,000$ replies (similar to the settings in Sec. B.2). While making predictions, the Hawkes process first simulates the main threads and then simulates

Model	NBA Dataset (MAE in Hrs)			
	1	5	10	20
Hawkes	$0.89_{0.82}$	$1.25_{1.12}$	$1.48_{1.31}$	$2.49_{1.68}$
NesTPP	$0.79_{0.68}$	$1.01_{0.89}$	$1.23_{0.84}$	$1.97_{1.37}$
SocialGrid-S	$0.79_{0.77}$	$1.11_{0.97}$	$1.27_{1.16}$	$1.93_{1.48}$
SocialGrid-M	$0.71_{0.71}$	$0.98_{0.84}$	$1.16_{1.06}$	$1.81_{1.27}$
SocialGrid	$0.71_{0.80}$	$0.99_{0.97}$	$1.17_{1.02}$	$1.87_{1.15}$

Model	NFL Dataset (MAE in Hrs)			
	1	5	10	20
Hawkes	$0.11_{0.09}$	$0.15_{0.12}$	$0.20_{0.15}$	$0.27_{0.21}$
NesTPP	$0.07_{0.05}$	$0.10_{0.08}$	$0.13_{0.11}$	$0.20_{0.17}$
SocialGrid-S	$0.06_{0.06}$	$0.12_{0.12}$	$0.13_{0.13}$	$0.22_{0.19}$
SocialGrid-M	$0.05_{0.07}$	$0.09_{0.11}$	$0.11_{0.11}$	$0.18_{0.19}$
SocialGrid	$0.05_{0.07}$	$0.10_{0.15}$	$0.12_{0.19}$	$0.19_{0.24}$

Table 3: **Adaptive Prediction of Main Threads.**

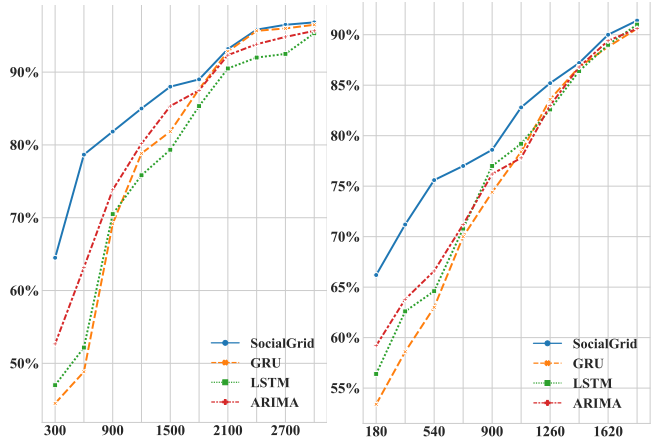
the replies under each predicted main thread. NesTPP [7] has been featured with a nested intensity function for dealing with the dynamic two-dimension sequences, and we follow the same setting therein. Note that the multivariate TPP like SEISMIC [8] is suitable for fix-dimensional sequence modeling and thus not applicable to our case. We randomly select the start point and simulate the arrival time of the following 20 main threads with the associated replies in the next 10 cells. This process was repeated for 20 times, and we report the average results in Tables 3 and 4.

Main Thread Arrival Time. As shown in Table 3, the average MAEs of all models exhibit an increasing trend over time, indicating that long-term adaptive prediction is a challenging task. Although the distinction of prediction errors among each approach is not far behind, the proposed framework SocialGrid and its variants still have the overall lowest MAE in both datasets. Our methods are generally more capable of learning both short- and long-range diffusion patterns by incorporating the influence of adjacent cascades. It is worth noting that NesTPP is comparable to SocialGrid in this experiment. In the ablation study, we see that the performance of SocialGrid-S is still the worst among the SocialGrid variants, and SocialGrid-M still offers the lowest prediction error.

Reply Number. Following the same procedure of simulating the main thread, given a start point, we predict the reply numbers in the following 2 to 10 time intervals, with d being 300 (resp., 150) second in the NBA (resp., NFL) dataset. According to Table 4, SocialGrid and its variant models have an overall considerable improvement in adaptively predicting the reply numbers, which suggests that mining the correlation between adjacent threads is critical for adaptive prediction.

Model	NBA Dataset ($d = 300$ sec)					NFL Dataset ($d = 180$ sec)				
	$2d$	$4d$	$6d$	$8d$	$10d$	$2d$	$4d$	$6d$	$8d$	$10d$
Hawkes	22.38 $_{6.32}$	19.78 $_{4.13}$	15.25 $_{4.12}$	12.38 $_{4.32}$	9.43 $_{2.38}$	17.98 $_{5.92}$	14.52 $_{3.79}$	11.13 $_{3.32}$	9.76 $_{3.12}$	7.72 $_{3.97}$
NesTPP	15.56 $_{5.62}$	13.86 $_{3.89}$	10.87 $_{6.12}$	8.59 $_{3.06}$	7.68 $_{2.79}$	12.82 $_{3.81}$	10.03 $_{3.65}$	9.15 $_{3.67}$	7.59 $_{2.61}$	5.89 $_{3.14}$
SocialGrid-S	5.98 $_{1.39}$	5.24 $_{1.13}$	4.14 $_{1.07}$	3.57 $_{0.98}$	2.93 $_{0.71}$	3.12 $_{0.69}$	2.98 $_{0.57}$	2.76 $_{0.48}$	2.63 $_{0.37}$	2.50 $_{0.31}$
SocialGrid-M	5.34 $_{1.07}$	4.65 $_{1.04}$	3.77 $_{0.96}$	2.87 $_{0.71}$	2.16 $_{0.47}$	2.90 $_{0.56}$	2.58 $_{0.39}$	2.47 $_{0.32}$	2.32 $_{0.36}$	2.28 $_{0.31}$
SocialGrid	5.27 $_{0.19}$	4.04 $_{0.83}$	3.26 $_{0.64}$	2.79 $_{0.51}$	1.87 $_{0.49}$	2.61 $_{0.58}$	2.49 $_{0.48}$	2.37 $_{0.42}$	2.28 $_{0.38}$	2.25 $_{0.35}$

Table 4: **Adaptive Prediction of Reply Numbers (MAE in reply numbers).**



(a) NBA (b) NFL

Fig. 2: **Breakout Threads Detection.** The x-axle denotes the length (sec) of the start duration, and the y-axle denotes the classification accuracy. There are 60 breakout cascades in the NBA dataset and 55 breakout cascades in the NFL dataset.

In contrast to what has been observed in Table 3, we see that there is a decreasing trend of MAE with the growth of the time intervals, and this is due plausibly to the *dying-out* characteristic [25]: the majority of the replies occur right after the posting time of the main thread. However, SocialGrid can better capture the historical impact from adjacent cascades to make a more accurate prediction. Finally, since SocialGrid is slightly better than its variants, the results indicate that the extra feature matrices are worthy of leveraging.

B.4. Application #3: Identifying Breakout Threads

While the previous sections have shown that SocialGrid performs better than the baselines, we cannot conclude that the arrival time of each individual event has been accurately predicted. A less ambitious task is to predict macro-level quantities such as the total replies of one thread, and one important application is to identify breakout threads, which is closely related to rumor detection and product advertisement in ODFs. More specifically, we aim at forecasting if a thread would attract more than average replies by using the data in the first several minutes. We first find the average cascade size among all cascades in our testing set, and define the breakout threads as those with a total number of replies more than twice the average. Given the information in the first kd seconds (called start duration) of a thread, $k \in \{1, ..., 10\}$, we employ the adaptive prediction approach to simulate the final volume of the replies and report if the thread would be classified as an outbreak cascade. The results measured by the classification accuracy are recorded in Fig 2. Intuitively, this task becomes easier when the information of a longer start duration is provided. As we can see from both figures, SocialGrid can identify more breakout threads in their early stages for both datasets. When the information of 300 seconds is provided, SocialGrid is able to identify roughly 65% of the breakout cascades in the NBA dataset; an accuracy of 66% is observed on the NFL dataset with a start duration of 180 seconds. Other approaches often need more than 600 seconds to achieve the same.

C. REFERENCES

[1] S. Goggins and W. Xing, "Building models explaining student participation behavior in asynchronous online discussion," *Computers & Education*, 2016.

[2] C. Romero, M.-I. López, J.-M. Luna, and S. Ventura, "Predicting students' final performance from participation in on-line discussion forums," *Computers & Education*, 2013.

[3] M. J. Thomas, "Learning within incoherent structures: The space of online discussion forums," *Journal of Computer Assisted Learning*.

[4] A. S. Lan, J. C. Spencer, Z. Chen, C. G. Brinton, and M. Chiang, "Personalized thread recommendation for mooc discussion forums," in *ECML*, 2018.

[5] H. Wang, C. Wang, C. Zhai, and J. Han, "Learning online discussion structures by conditional random fields," in *SIGIR*, 2011.

[6] A. N. Medvedev, J.-C. Delvenne, and R. Lambiotte,

"Modelling structure and predicting dynamics of discussion threads in online boards," *Journal of Complex Networks*, 2019.

[7] C. Ling, G. Tong, and M. Chen, "Nestpp: Modeling thread dynamics in online discussion forums," in *HT*, 2020.

[8] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *SIGKDD*, 2015.

[9] P. Bao, H.-W. Shen, X. Jin, and X.-Q. Cheng, "Modeling and predicting popularity dynamics of microblogs using self-excited hawkes processes," in *WWW*, 2015.

[10] R. Kobayashi and R. Lambiotte, "Tideh: Time-dependent hawkes process for predicting retweet dynamics," in *ICWSM*, 2016.

[11] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, 1971.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv:1409.0473*.

[13] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv:1508.04025*, 2015.

[14] G. E. Hinton, "Connectionist learning procedures," in *Machine learning*.

[15] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce english text," *Complex systems*, 1987.

[16] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *arXiv:1603.06995*.

[17] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *IJCNN*.

[18] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[19] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[20] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "Arima models to predict next-day electricity prices," *IEEE transactions on power systems*, 2003.

[21] "Python library," https://www.statsmodels.org.

[22] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, 2002.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv:1412.3555*, 2014.

[24] "Hawkeslib," pypi.org/project/hawkeslib/.

[25] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted?" in *WWW*, 2014.